

Contents

1 Routine/Function Prologues	2
1.0.1 getgdas.F90 (Source File: getgdas.F90)	2
1.1 Core Functions of getgdas	2
1.1.1 gdasfile (Source File: getgdas.F90)	9
1.1.2 gdasfilef06 (Source File: getgdas.F90)	11

1 Routine/Function Prologues

1.0.1 **getgdas.F90** (Source File: *getgdas.F90*)

Opens, reads, and interpolates NCEP-GDAS forcing.

TIME1 = most recent past data

TIME2 = nearest future data

The idea is to open either the 00 or 03 forecast file associated with the most recent GDAS assimilation (available every 6 hours). Precipitation rates and radiation fluxes will be taken from the F03 and F06 files, since averages are provided.

- if that fails, the strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day.

1.1 Core Functions of **getgdas**

tick Determines GDAS data times

gdasfile Puts together appropriate file name for 3 hour intervals

gdasfilef06 Puts together appropriate file name for 6 hour intervals

retgdas Interpolates GDAS data to LDAS grid

REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
25 Oct 1999: Jared Entin; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Fixed name construction error
              in Subroutine ETA6HFILE
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
              data with opposite sign convention from recent shortwave data.
              Added capability to use time averaged shortwave and longwave data.
              Altered times which are passed into ZTERP--used to be GMT1 and GMT2,
              now they are LDAS%ETATIME1 and LDAS%ETATIME2
11 May 2000: Brian Cosgrove; Added checks for SW values that are too high
              due to zenith angle weighting...if too high, use linear weighting.
              Also, if cos(zen) less than .1, then use linear weighting to
              avoid computed values of SW that are too high.
18 May 2000: Brian Cosgrove; Corrected line of code in ETAEDASNAME which
              assigned wrong year directory variable when constructing
              EDAS filename
26 May 2000: Jared Entin; Changed numerical bound of the TRY variable
              to fix a rollback problem.
5 June 2000: Brian Cosgrove; Fixed a problem with the correction of the negative
              radiation sign convention. Prior to fix, was not correcting negative
              values of -999.9...now it changes all negative values to positive ones.
18 Aug 2000: Brian Cosgrove; Fixed undefined value problem in check for negative
              radiation values over land points.
08 Dec 2000: Urszula Jambor; Rewrote geteta.f in fortran90 to use GDAS in GLDAS

```

15 Mar 2001: Jon Gottschalck; Slight change to handle more forcing parameters at time step 0.
 09 Apr 2001: Urszula Jambor; Added capability of using DAAC forcing data every 6 hours, rather than every 3 hours.
 30 May 2001: Urszula Jambor; Changed forcing used: T,q,u fields taken from F00 & F03 files, radiation and precip. fields taken from F06 & F03 (F03 fields are subtracted out from F06)

INTERFACE:

```
subroutine getgdas()
```

USES:

```
use lisdrv_module, only : lis, gindex
use baseforcing_module, only: glbdata1, glbdata2
use time_manager
use time_module, only : tick
use gdasdomain_module, only : gdasdrv
use def_ipMod, only : def_ip_input
use spmdMod
#if ( defined OPENDAP )
  use opendap_module, only : parm_nc, parm_nr, nroffset
#endif
```

CONTENTS:

```
lis%f%FOO_flag = 0
lis%f%F06_flag = 0
lis%f%findtime1=0
lis%f%findtime2=0
movetime=0
!-----
! Determine the correct number of forcing variables
!-----
if ( masterproc ) then
  nstep = get_nstep()
endif
#if ( defined OPENDAP )
  call MPI_BCAST(nstep,1,MPI_INTEGER,0,MPI_COMM_WORLD,ferror)
#endif
if( nstep == 0 ) then
  nforce = gdasdrv%nmif
else
  nforce = lis%f%nf
endif
if ( nstep == 1 .or. lis%f%rstflag == 1) then
  lis%f%findtime1=1
  lis%f%findtime2=1
  glbdata1 = 0
```

```
glbdata2 = 0
movetime=0           ! movetime is not properly set at time-step = 1
lis%f%rstflag = 0
endif
!-----
! Determine required GDAS data times
! (previous assimilation, current & future assimilation hours)
! The adjustment of the hour and the direction will be done
! in the subroutines that generate the names
!-----
yr1 = lis%t%yr  !current time
mo1 = lis%t%mo
da1 = lis%t%da
hr1 = lis%t%hr
mn1 = lis%t%mn
ss1 = 0
ts1 = 0
call tick( timenow, doy1, gmt1, yr1, mo1, da1, hr1, mn1, ss1, ts1 )

yr1 = lis%t%yr  !previous assimilation/forecast hour
mo1 = lis%t%mo
da1 = lis%t%da
if ( lis%f%F06_flag == 0 ) then
    hr1 = 3*(int(real(lis%t%hr)/3.0))
else
    hr1 = 6*(int(real(lis%t%hr)/6.))
end if
mn1 = 0
ss1 = 0
ts1 = 0
call tick( time1, doy1, gmt1, yr1, mo1, da1, hr1, mn1, ss1, ts1 )

yr2 = lis%t%yr  !next assimilation/forecast hour
mo2 = lis%t%mo
da2 = lis%t%da
if ( lis%f%F06_flag == 0 ) then
    hr2 = 3*(int(real(lis%t%hr)/3.0))
else
    hr2 = 6*(int(real(lis%t%hr)/6.0))
end if
mn2 = 0
ss2 = 0
if ( lis%f%F06_flag == 0 ) then
    ts2 = 3*60*60
else
    ts2 = 6*60*60
end if
call tick( time2, doy2, gmt2, yr2, mo2, da2, hr2, mn2, ss2, ts2 )
```

```
!-----
! Use these if need to roll back time.
!-----
dumbtime1 = time1
dumbtime2 = time2
!-----
! Check to see if current time (timenow) has crossed past gdastime2,
! requiring that both gdastime2 be assigned to gdastime1 and a new
! gdastime2 be set 3 or 6 hours ahead of the current gdastime2.
!-----
if ( timenow > gdasdrv%gdastime2 ) then
    movetime = 1
    lis%f%findtime2 = 1
end if

if(time1>gdasdrv%griduptime1 .and. &
    time1 <gdasdrv%griduptime2.and. &
    gdasdrv%gridchange1) then
    print*, 'changing grid....to 2000-2002'

    gdasdrv%ncold = 512
    gdasdrv%nrold = 256
!-----
! Reinitialize the weights and neighbors
!-----
kgdsi = 0
kgdsi(1) = 4
kgdsi(2) = 512
kgdsi(3) = 256
kgdsi(4) = 89463
kgdsi(5) = 0
kgdsi(6) = 128
kgdsi(7) = -89463
kgdsi(8) = -703
kgdsi(9) = 703
kgdsi(10) = 128
kgdsi(20) = 255
call def_ip_input(kgdsi)
gdasdrv%gridchange1 = .false.
else if(time1 > gdasdrv%griduptime2.and. &
        gdasdrv%gridchange2) then
    print*, 'changing grid to 2002-'
    gdasdrv%ncold = 768
    gdasdrv%nrold = 384
!-----
! Reinitialize the weights and neighbors
!-----
kgdsi = 0
```

```

kgdsi(1) = 4
kgdsi(2) = 768
kgdsi(3) = 384
kgdsi(4) = 89462
kgdsi(5) = 0
kgdsi(6) = 128
kgdsi(7) = -89462
kgdsi(8) = -469
kgdsi(9) = 469
kgdsi(10) = 192
kgdsi(20) = 255
call def_ip_input(kgdsi)
gdasdrv%gridchange2 = .false.
endif
!-----
! Establish gdastime1
!-----
if ( lis%f%findtime1 == 1 ) then !get new time1 from the past
  print *, 'Getting new time1 data'
  ferror = 0
  order = 1
  try = 0
  ts1 = -24*60*60

  do
    if ( ferror /= 0 ) exit
    try = try+1
    if ( lis%f%F06_flag == 0 ) then
      call gdasfile   ( name, gdasdrv%gdasdir, yr1, mo1, da1, hr1 )
    else
      call gdasfileF06( name, gdasdrv%gdasdir, yr1, mo1, da1, hr1 )
    end if
    print*, 'Reading GDAS file1 ',name
    !call retgdas( order, lis, gindex, name, nameF06, 0,ferror, try)
    call retgdas( order, lis, name, nameF06, 0,ferror, try)

    if ( ferror == 1 ) then
!-----
! successfully retrieved forcing data
!-----
      gdasdrv%gdastime1 = time1
    else
!-----
! ferror still=0, so roll back one day & start again
!-----
      call tick( dumptime1, doy1, gmt1, yr1, mo1, da1, hr1, mn1, ss1, ts1 )
    end if
    if ( try > ndays ) then

```

```
!-----
! data gap exceeds 10 days so stop
!-----
      print *, 'ERROR: GDAS data gap exceeds 10 days on file 1'
      call endrun
    end if
  end do
end if
!-----
! Establish gdastime2
!-----
if ( movetime == 1 ) then
  gdasdrv%gdastime1 = gdasdrv%gdastime2
  lis%f%findtime2 = 1
  do f = 1, nforce
    do c = 1, lis%d%ngrid
      glbdata1(f,c) = glbdata2(f,c)
    end do
  end do
end if

if ( lis%f%findtime2 == 1 ) then
  print *, 'Getting new time2 data'
  lis%f%F00_flag = 0

  ferror = 0
  order = 2
  try = 0
  ts2 = -24*60*60
!-----
! determine if both F00 & F06 files needed
!-----
if ( lis%f%F06_flag == 0 ) then
  if ( modulo(hr2,2) > 0 ) then
    lis%f%F00_flag = 0 !odd hr, use F03 file
  else
    lis%f%F00_flag = 1 !even hr, use F00 & F06
  end if
end if

do
  if ( ferror /= 0 ) exit
  try = try+1
  if ( lis%f%F06_flag == 0 ) then
    if ( lis%f%F00_flag == 0 ) then
      call gdasfile( name, gdasdrv%gdasdir, yr2, mo2, da2, hr2 )
    else
      call gdasfile( name, gdasdrv%gdasdir, yr2, mo2, da2, hr2 )
```

```

        call gdasfileF06( nameF06, gdasdrv%gdasdir, yr2, mo2, da2, hr2 )
    end if
else
    call gdasfileF06( name, gdasdrv%gdasdir, yr2, mo2, da2, hr2 )
end if
print*, 'Reading GDAS file2 ',name
!call retgdas( order, lis, gindex, name, nameF06, &
!    lis%f%FOO_flag, ferror,try )
call retgdas( order, lis, name, nameF06, &
    lis%f%FOO_flag, ferror,try )
if ( ferror == 1 ) then
!---
! successfully retrieved forcing data
!---
        gdasdrv%gdastime2 = time2
else
!---
! ferror still=0, so roll back one day & start again
!---
        call tick( dumptime2, doy2, gmt2, yr2, mo2, da2, hr2, mn2, ss2, ts2 )
    end if
    if ( try > ndays ) then
!---
! data gap exceeds 10 days so stop
!---
        print *, 'ERROR: GDAS data gap exceeds 10 days on file 2'
        call endrun
    end if
    end do
end if
!---
! loop through all forcing parameters & interpolate
!---
do f = 1, lis%f%nforce
    if ( (f == 3) .or. (f == 4) ) then
        do c = 1, nc_dom !c = 1, lis%d%lnc
            do r = 1+nroffset, nr_dom+nroffset !r = 1, lis%d%lnr
                index = gindex(c,r)
                if(index .ne.-1) then
                    if ( (glbdata2(f,index) /= -9999.9) .and. &
                        (glbdata2(f,index) < 0)) then
                        glbdata2(f,index) = (-1) * glbdata2(f,index)
                    end if
                    if ( (glbdata1(f,index) /= -9999.9) .and. &
                        (glbdata1(f,index) < 0)) then
                        glbdata1(f,index) = (-1) * glbdata1(f,index)
                    end if
                end if
            end do
        end do
    end if
end do

```

```

        end do
    end do
end if
enddo
```

1.1.1 gdasfile (Source File: getgdas.F90)

This subroutine puts together GDAS file name for 3 hour file intervals

INTERFACE:

```
subroutine gdasfile( name, gdasdir, yr, mo, da, hr )  
  
    implicit none
```

INPUT PARAMETERS:

```
character(len=80) :: gdasdir  
integer :: yr, mo, da, hr
```

OUTPUT PARAMETERS:

```
character(len=80) :: name
```

CONTENTS:

```

!-----
! if hour is 00 or 03, look for 00ZF00 or 00ZF03
! if hour is 06 or 09, look for 06ZF00 or 06ZF03
! if hour is 12 or 15, look for 12ZF00 or 12ZF03
! if hour is 18 or 21, look for 18ZF00 or 18ZF03
!-----

      if ( uhr <= 3 ) then
          initcode = '00'
      else if (uhr <= 9 ) then
          initcode = '06'
      else if ( uhr <= 15 ) then
          initcode = '12'
      else if ( uhr <= 21 ) then
          initcode = '18'
      end if
      if ( remainder > 0 ) then
          fcstcode = '03'
      else
          fcstcode = '00'
      end if

      write(UNIT=temp, fmt='(a40)') gdasdir
      read(UNIT=temp, fmt='(80a1)') (fbase(i), i=1,80)

      write(UNIT=temp, fmt='(a1, i4, i2, a1)') '/', uyr, umo, '/'
      read(UNIT=temp, fmt='(8a1)') fdir
      do i = 1, 8
          if ( fdir(i) == ' ' ) fdir(i) = '0'
      end do

      write(UNIT=temp, fmt='(i4, i2, i2, a2)') uyr, umo, uda, initcode
      read(UNIT=temp, fmt='(10a1)') ftime
      do i = 1, 10
          if ( ftime(i) == ' ' ) ftime(i) = '0'
      end do

      write(UNIT=temp, fmt='(a16, a2, a3)') '.gdas1.sfluxgrbf', fcstcode, '.sg'
      read (UNIT=temp, fmt='(80a1)') (fsubs(i), i=1,21)

      c = 0
      do i = 1, 80
          if ( (fbase(i) == ' ') .and. (c == 0) ) c = i-1
      end do

      write(UNIT=temp, fmt='(80a1)') (fbase(i), i=1,c), (fdir(i), i=1,8), &
                                     (ftime(i), i=1,10), (fsubs(i), i=1,21)

      read(UNIT=temp, fmt='(a80)') name

```

```
return
```

1.1.2 gdasfilef06 (Source File: getgdas.F90)

This subroutine puts together GDAS file name for 6 hour forecast files (DAAC archive friendly)

INTERFACE:

```
subroutine gdasfileF06( name, gdasdir, yr, mo, da, hr )
```

USES:

```
use time_module
implicit none
```

INPUT PARAMETERS:

```
character(len=80) :: gdasdir
integer :: yr, mo, da, hr
```

OUTPUT PARAMETERS:

```
character(len=80) :: name
```

CONTENTS:

```
92 format (80a1)
93 format (a80)
94 format (i4, i2, i2, a2)
95 format (10a1)
96 format (a40)
97 format (a16, a2, a3)
98 format (a1, i4, i2, a1)
99 format (8a1)
!-----
! Make variables for the time used to create the file
! We don't want these variables being passed out
!-----
uyr = yr
umo = mo
uda = da
uhr = 6*(hr/6) !hour needs to be a multiple of 6 hours
umn = 0
uss = 0
ts1 = -24*60*60 !one day interval to roll back date.
```

```

!-----
! if hour is 00Z look for 18ZF06 after rolling back one day
! if hour is 06Z look for 00ZF06
! if hour is 12Z look for 06ZF06
! if hour is 18Z look for 12ZF06
!-----

fcstcode = '06'
if ( uhr == 0 ) then
  call tick( dumbtime, doy, gmt, uyr, umo, uda, uhr, umn, uss, ts1 )
  initcode = '18'
else if ( uhr == 6 ) then
  initcode = '00'
else if ( uhr == 12 ) then
  initcode = '06'
else if ( uhr == 18 ) then
  initcode = '12'
end if

write(UNIT=temp, fmt='(a40)') gdasdir
read(UNIT=temp, fmt='(80a1)') (fbase(i), i=1,80)

write(UNIT=temp, fmt='(a1, i4, i2, a1)') '/', uyr, umo, '/'
read(UNIT=temp, fmt='(8a1)') fdir
do i = 1, 8
  if ( fdir(i) == ' ' ) fdir(i) = '0'
end do

write(UNIT=temp, fmt='(i4, i2, i2, a2)') uyr, umo, uda, initcode
read(UNIT=temp, fmt='(10a1)') ftime
do i = 1, 10
  if ( ftime(i) == ' ' ) ftime(i) = '0'
end do
write(UNIT=temp, fmt='(a16, a2, a3)') '.gdas1.sfluxgrbf', '06', '.sg'
read (UNIT=temp, fmt='(80a1)') (fsubs(i), i=1,21)
c = 0
do i = 1, 80
  if ( (fbase(i) == ' ') .and. (c == 0) ) c = i-1
end do

write(UNIT=temp, fmt='(80a1)') (fbase(i), i=1,c), (fdir(i), i=1,8), &
                           (ftime(i), i=1,10), (fsubs(i), i=1,21)

read(UNIT=temp, fmt='(a80)') name
return

```